Requested Patent: JP2003229000A

Title: METHOD AND APPARATUS FOR MEMORY SELF TESTING ;

Abstracted Patent: US2003120985 ;

Publication Date: 2003-06-26 ;

Inventor(s):

HILL STEPHEN JOHN (US); SLOBODNIK RICHARD (US); WILLIAMS GERARD RICHARD (US) ;

Applicant(s): ;

Application Number: US20010025816 20011226 ;

Priority Number(s): US20010025816 20011226 ;

IPC Classification: G11C29/00 ;

Equivalents: GB2383640 ;

ABSTRACT:

A self-test controller 10 is responsive to scanned in self-test instructions to carry out test operations including generating a sequence of memory addresses that is specified by the self-test instruction. Combining multiple such self-test instructions allows a custom test methodology to be built up by a user using a generic self-test controller 10

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2003－229000
（P2003－229000A）
(43)公開日　平成15年８月15日(2003.8.15)

(51)Int.Cl.'		識別記号	FI		テーマコード(参考)	
G11C	29/00	675	G11C	29/00	675M	2G132
G01R	31/28		G06F	12/16	330A	5B018
G06F	12/16	330		15/78	510K	5B062
	15/78	510	G01R	31/28	U	5L106
					B	

審査請求　未請求　請求項の数34　ＯＬ　（全 18 頁）

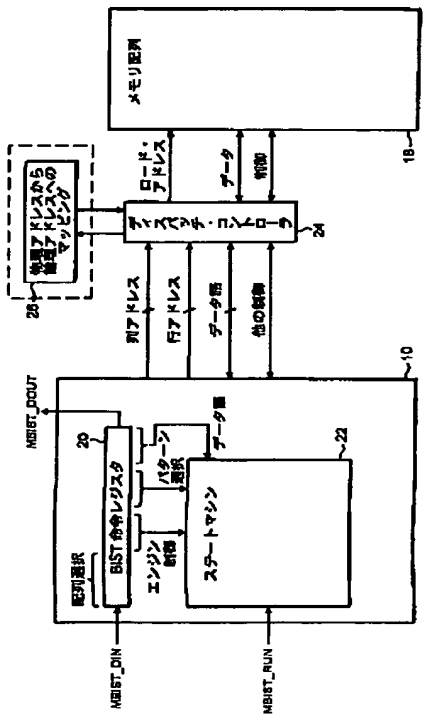
(21)出願番号　特願2002－288612(P2002－288612)
(22)出願日　平成14年10月１日(2002.10.1)
(31)優先権主張番号　０２５８１６
(32)優先日　平成13年12月26日(2001.12.26)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　594154428
エイアールエム　リミテッド
イギリス国　シーピー１　９エヌジェイ
ケンブリッジ，チェリー　ヒントン，フル
パーン　ロード　110
(72)発明者　リチャード　スロボドニク
アメリカ合衆国　テキサス、オースティ
ン、　ビル　ボニー　コート　108
(74)代理人　100066692
弁理士　浅村　皓　（外３名）

最終頁に続く

(54)【発明の名称】　メモリ自己テストの方法と装置

(57)【要約】
【課題】　メーカや設計方式が異なる種々のメモリの不良を自己テストするのに有用な、プログラム可能な自己テスト・コントローラを提供する。
【解決手段】　自己テスト・コントローラ１０は取り込まれた自己テスト命令に応答して、自己テスト命令が指定する、一連のメモリ・アドレスの生成を含むテスト動作を行う。多数のこのような自己テスト命令を組み合わせることにより、ユーザは総称的自己テスト・コントローラ１０を用いて特注のテスト方法を構築できる。



【特許請求の範囲】

【請求項１】　データ処理装置であって、
それぞれのメモリ・アドレスに関連する複数のメモリ記憶位置を有する少なくとも１個のメモリと、
前記少なくとも１個のメモリの自己テストを制御する自己テスト・コントローラと、を含み、
前記自己テスト・コントローラは自己テスト命令に応答して一連のメモリ記憶位置内の各メモリ位置に少なくとも一度メモリ・アクセスを行い、前記自己テスト命令により選択された前記一連のメモリ記憶位置内で順次にアクセスされるメモリ位置に従ってメモリ・アドレスは変化し、前記自己テスト・コントローラは前記自己テスト命令により形成されて異なるメモリ・テスト方法を実現する、データ処理装置。

【請求項２】　前記自己テスト・コントローラは複数の自己テスト命令を実行して一連のメモリ・テストを行い、前記自己テスト命令はプログラム可能であって異なるユーザはメモリ・テストの異なるシーケンスを指定できる、請求項１記載のデータ処理装置。

【請求項３】　前記メモリ・テストのシーケンスは異なるメモリに適合するように変更できる、請求項２記載のデータ処理装置。

【請求項４】　前記メモリ・テストのシーケンスは異なる製作特性とテスト要求に適合するように変更できる、請求項２記載のデータ処理装置。

【請求項５】　前記メモリ・アドレスは前記少なくとも１個のメモリ内の物理的行アドレスと列アドレスである、請求項１記載のデータ処理装置。

【請求項６】　前記自己テスト命令は前記少なくとも１つのメモリ・アクセスの一部として前記メモリに書き込むデータを指定する、請求項１記載のデータ処理装置。

【請求項７】　前記メモリ・アドレスは前記自己テスト命令により選択された順次のメモリ位置に従って変化し、また前記自己テスト・コントローラは次の１つ以上のメモリ・テスト動作を実行する、すなわち、(i)　指定されたデータを或る範囲内のメモリ・アドレスの全てのメモリ位置に書き込み、(ii)　データを或る範囲内のメモリ・アドレスの全てのメモリ位置から読み取り、(iii)　指定されたデータをチェッカー盤パターンのメモリ・アドレスを有するメモリ位置に書き込み、(iv)　データをチェッカー盤パターンのメモリ・アドレスを有するメモリ位置から読み取り、(v)　マーチCメモリ・テストを行い、(vi)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取りまた指定されたデータを書き込み、メモリ位置の或る行内のメモリ位置に順にアクセスした後でメモリ位置の次の行を選択してアクセスし、(vii)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取りまた指定されたデータを書き込み、メモリ位置の或る列内のメモリ位置に順にアクセスした後でメモリ位置の次の列を選択してアクセスし、(viii)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取り、指定されたデータを書き込み、またデータを読み取り、メモリ位置の或る行内のメモリ位置に順にアクセスした後でメモリ位置の次の行を選択してアクセスし、(ix)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取り、指定されたデータを書き込み、またデータを読み取り、メモリ位置の或る列内のメモリ位置に順にアクセスした後でメモリ位置の次の列を選択してアクセスし、(x)　一連のメモリ位置において、前記メモリ内の１つ以上のビット線に或る値を繰り返し書き込み、次に前記１つまたは複数のビット線を共用するメモリ位置内に記憶されている相補値を読み取り、(xi)　一連のメモリ位置において、或るメモリ位置から或る値を繰り返し読み取ると共に逆のデータ書込みを行い、(xii)　メーカのテスト方法が特定の要求を持たない実行/不実行テストについて(i)から(xi)に規定された所定の組み合わせのメモリ・テスト動作を行い、(xiii)　不合格検出を有効化するために特定の点で偽読取りデータを生成する請求項１記載のデータ処理装置。

【請求項８】　プロセッサ・コアを更に備え、前記プロセッサ・コアと前記少なくとも１個のメモリと前記自己テスト・コントローラを１つの集積回路上に形成する、請求項１記載のデータ処理装置。

【請求項９】　前記少なくとも１個のメモリは合成メモリか特注メモリのどちらかである、請求項１記載のデータ処理装置。

【請求項１０】　インターフェース回路を前記自己テスト・コントローラと前記少なくとも１個のメモリの間に設け、前記インターフェース回路を通して信号の値とタイミングを前記自己テスト・コントローラと前記少なくとも１個のメモリの間で送って前記少なくとも１個のメモリの異なる値およびタイミング特性に対処する、請求項１記載のデータ処理装置。

【請求項１１】　前記インターフェース回路は前記自己テスト・コントローラが生成したメモリ・アドレス値を、前記少なくとも１個のメモリに入力する論理アドレス値にマッピングする、請求項１０記載のデータ処理装置。

【請求項１２】　複数のメモリを備え、前記自己テスト命令は前記自己テスト命令を前記複数のメモリのどれに与えるかを指定する、請求項１記載のデータ処理装置。

【請求項１３】　前記自己テスト命令は、検出されたメモリ誤りを前記自己テスト・コントローラが複数の異なる方法のどれを用いて報告するかを指定する、請求項１記載のデータ処理装置。

【請求項１４】　前記インターフェース回路は前記少なくとも１個のメモリのテストから得られたデータを収集する結果データ・レジスタを含み、また前記自己テスト

・コントローラは自己テスト命令に応じて結果データを前記結果データ・レジスタから読み取る、請求項１０記載のデータ処理装置。

【請求項１５】　前記自己テスト命令はテスト対象の前記少なくとも１個のメモリのサイズを指定する、請求項１記載のデータ処理装置。

【請求項１６】　前記自己テスト命令は前記自己テスト・コントローラに直列にロードされる、請求項１記載のデータ処理装置。

【請求項１７】　前記少なくとも１個のメモリと前記自己テスト・コントローラを複数個の外部信号ピンを有する集積回路上に形成し、前記自己テスト・コントローラは１つ以上の自己テスト命令を前記自己テスト・コントローラに与えるのに用いる１個以上の外部信号ピンを有する、請求項１記載のデータ処理装置。

【請求項１８】　それぞれのメモリ・アドレスに関連する複数のメモリ記憶位置を有するメモリ・テスト方法であって、前記メモリに結合する自己テスト・コントローラに自己テスト命令を与え、前記自己テスト命令に応答して一連のメモリ記憶位置内の各メモリ位置に少なくとも一度メモリ・アクセスを行い、前記自己テスト命令により選択された前記一連のメモリ記憶位置内で順次にアクセスされるメモリ位置に従ってメモリ・アドレスは変化し、前記自己テスト・コントローラは前記自己テスト命令により形成されて異なるメモリ・テスト方法を実現する、ステップを含む、メモリ・テスト方法。

【請求項１９】　前記自己テスト・コントローラで複数の自己テスト命令を実行して一連のメモリ・テストを行い、前記自己テスト命令はプログラム可能であって異なるユーザはメモリ・テストの異なるシーケンスを指定できる、請求項１８記載のメモリ・テスト方法。

【請求項２０】　前記メモリ・テストのシーケンスは異なるメモリに適合するように変更できる、請求項１９記載のメモリ・テスト方法。

【請求項２１】　前記メモリ・テストのシーケンスは異なる製作特性とテスト要求に適合するように変更できる、請求項１９記載のメモリ・テスト方法。

【請求項２２】　前記メモリ・アドレスは前記少なくとも１個のメモリ内の物理的行アドレスと列アドレスである、請求項１８記載のメモリ・テスト方法。

【請求項２３】　前記自己テスト命令は前記少なくとも１つのメモリ・アクセスの一部として前記メモリに書き込むデータを指定する、請求項１８記載のメモリ・テスト方法。

【請求項２４】　前記メモリ・アドレスは前記自己テスト命令により選択された順次のメモリ位置に従って変化し、また前記自己テスト・コントローラは次の１つ以上のメモリ・テスト動作を実行する、すなわち、(i)　指定されたデータを或る範囲内のメモリ・アドレスの全てのメモリ位置に書き込み、(ii)　データを或る範囲内のメモリ・アドレスの全てのメモリ位置から読み取り、(iii)　指定されたデータをチェッカー盤パターンのメモリ・アドレスを有するメモリ位置に書き込み、(iv)　データをチェッカー盤パターンのメモリ・アドレスを有するメモリ位置から読み取り、(v)　マーチCメモリ・テストを行い、(vi)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取りまた指定されたデータを書き込み、メモリ位置の或る行内のメモリ位置に順にアクセスした後でメモリ位置の次の行を選択してアクセスし、(vii)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取りまた指定されたデータを書き込み、メモリ位置の或る列内のメモリ位置に順にアクセスした後でメモリ位置の次の列を選択してアクセスし、(viii)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取り、指定されたデータを書き込み、またデータを読み取り、メモリ位置の或る列内のメモリ位置に順にアクセスした後でメモリ位置の次の列を選択してアクセスし、(ix)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取り、指定されたデータを書き込み、またデータを読み取り、メモリ位置の或る列内のメモリ位置に順にアクセスした後でメモリ位置の次の列を選択してアクセスし、(x)　一連のメモリ位置において、前記メモリ内の１つまたは複数のビット線に或る値を繰り返し書き込み、次に前記１つ以上のビット線を共用するメモリ位置内に記憶されている相補値を読み取り、(xi)　一連のメモリ位置において、或るメモリ位置から或る値を繰り返し読み取ると共に逆のデータ書込みを行い、(xii)　メーカのテスト方法が特定の要求を持たない実行/不実行テストについて(i)から(xi)に規定された所定の組み合わせのメモリ・テスト動作を行い、(xiii)　不合格検出を確認するために特定の点で偽読取りデータを生成する請求項１８記載のメモリ・テスト方法。

【請求項２５】　前記少なくとも１個のメモリと前記自己テスト・コントローラを１つの集積回路上に形成する、請求項１８記載のメモリ・テスト方法。

【請求項２６】　前記少なくとも１個のメモリは合成メモリと特注メモリのどちらかである、請求項１８記載のメモリ・テスト方法。

【請求項２７】　前記自己テスト・コントローラと前記少なくとも１個のメモリの間に送られる信号の値とタイミングは前記自己テスト・コントローラと前記少なくとも１個のメモリの間に設けられたインターフェース回路により処理され、前記少なくとも１個のメモリの異なる値およびタイミング特性に対処する、請求項１８記載のメモリ・テスト方法。

【請求項２８】　前記インターフェース回路は前記自己テスト・コントローラが生成したメモリ・アドレス値を、前記少なくとも１個のメモリに入力する論理アドレ

ス値にマッピングする、請求項２７記載のメモリ・テスト方法。
【請求項２９】　前記自己テスト命令は前記自己テスト命令を前記複数のメモリのどれに与えるかを指定する、請求項１８記載のメモリ・テスト方法。
【請求項３０】　前記自己テスト命令は、検出されたメモリ誤りを前記自己テスト・コントローラが複数の異なる方法のどれを用いて報告するかを指定する、請求項１８記載のメモリ・テスト方法。
【請求項３１】　前記インターフェース回路は前記少なくとも１個のメモリのテストから得られたデータを収集する結果データ・レジスタを含み、また前記自己テスト・コントローラは自己テスト命令に応答して結果データを前記結果データ・レジスタから読み取る、請求項２７記載のメモリ・テスト方法。
【請求項３２】　前記自己テスト命令はテスト対象の前記少なくとも１個のメモリのサイズを指定する、請求項１８記載のメモリ・テスト方法。
【請求項３３】　前記自己テスト命令は前記自己テスト・コントローラに直列にロードされる、請求項１８記載のメモリ・テスト方法。
【請求項３４】　前記少なくとも１個のメモリと前記自己テスト・コントローラを複数個の外部信号ピンを有する集積回路上に形成し、前記自己テスト・コントローラは１つ以上の自己テスト命令を前記自己テスト・コントローラに与えるのに用いる１個以上の外部信号ピンを有する、請求項１８記載のメモリ・テスト方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明はデータ処理装置の分野に関する。より詳しくは、本発明はメモリ不良を検出するための、データ処理装置内のメモリの自己テストに関する。
【０００２】
【従来の技術】自己テスト機構（組込み自己テスト（ＢＩＳＴ）と呼ぶことがある）を持つメモリを組み込んだデータ処理装置は公知である。自己テスト機構とは、メモリ回路を製作したときに一度または一連の自己テストを行ってメモリ不良が存在するかどうかを判定し、存在する場合はその回路を除外することを指示する機構をいう。メモリ回路は高い信頼性が要求されるので、不良回路は正しく識別しなければならない。メモリ・サイズが大きくなるに従って、メモリが占めるダイの部分も大きくなる。またメモリ内のトランジスタの密度が高くなるとメモリの不良密度も非常に大きくなる。
【０００３】
【発明が解決しようとする課題】不良の種類は設計によって大幅に異なり、また同じ設計でも製造プロセスによって異なる。このため、通常、自己テスト方式は設計や製造方法に従ってメーカが選択する。一般にメーカは、特定の環境において不良メモリを識別するのに適した、経験から得られた自社特有の好ましいテスト方法を有する。或るメーカが採用しているテスト方法は他のメーカにとって必ずしも適当な方法ではない。
【０００４】自己テスト回路はメモリが正しく製作されていることを確かめるのに必要であるが、或る回路領域を占有するのでできるだけ小さくすることが望ましい。自己テスト回路は一般に集積回路を製造するときに用いるが、その後集積回路の通常の動作中には用いない。
【０００５】
【課題を解決するための手段】１つの形態では、本発明はデータを処理する装置を提供する。この装置は、それぞれのメモリ・アドレスに関連する複数のメモリ記憶位置を有する少なくとも１個のメモリと、前記少なくとも１個のメモリの自己テストを制御する自己テスト・コントローラと、を含み、前記自己テスト・コントローラは自己テスト命令に応じて一連のメモリ記憶位置内の各メモリ位置に少なくとも一度メモリ・アクセスを行い、前記自己テスト命令により選択された前記一連のメモリ記憶位置内で順次にアクセスされるメモリ位置に従ってメモリ・アドレスは変化し、前記自己テスト・コントローラは前記自己テスト命令により形成されて異なるメモリ・テスト方法を実現する。
【０００６】本発明の認識では、アドレス指定の順序を変えることにより種々のテスト方法を行うことができるプログラム可能な自己テスト・コントローラが望ましい場合がある。詳しく述べると、かかるプログラム可能な自己テスト・コントローラを提供することにより、多くの異なるメーカがこの自己テスト・コントローラの設計を再使用することが可能になり、しかも各メーカがこの自己テスト・コントローラをプログラムして、自分の特定の環境に最も適したものとして選択したテスト方法を実行することが可能になる。単一の固定されたテスト方法だけを行うコントローラとは異なり、プログラム可能な自己テスト・コントローラを用いると自己テスト・コントローラのサイズと複雑さが増加するので一般に不利と考えられるが、自己テスト・コントローラ設計を再使用すれば、異なるメーカの種々のテスト要求や、同じメーカであっても独自の製造プロセスを開発したときに生じる異なるテスト要求などに対処できるので、この不利を十分補うことができる。
【０００７】自己テスト命令が完結したテスト方法を示すことも可能であるが、本発明の好ましい実施の形態では、複数の自己テスト命令を実行して、異なるユーザが異なるシーケンスを指定するように調整された一連のメモリ・テストを行う。自己テスト・コントローラは自己テスト命令のプログラムにより駆動され、特定のユーザが望むあらゆるテスト方法を実行する。理解されるように、テスト対象のメモリの特定の種類に適合するように、または特定の製造テスト方法に適合するように、一

連のメモリ・テストを調整できる。

【0008】理解されるように、自己テスト・コントローラが作成するメモリ・アドレスは種々の形をとってよく、また種々の方法でメモリ位置にマッピングしてよい。しかし多くの種類のメモリ不良はメモリ位置自体の、または他のメモリ位置に対する相対的な物理的位置に関連するので、自己テスト・コントローラが作成するメモリ・アドレスはメモリ位置の物理的位置を与える行と列のアドレスを直接表すものが特に適している。なぜなら、一般にテストの際に有用なメモリ・アドレスのシーケンスは、メモリ・アドレスをこの方法で表現することにより容易に形成できるからである。

【0009】かかるプログラム可能な自己テスト・コントローラにおいて考慮すべき重要な点は、どのアドレス・シーケンスとどのテストを命令が支援するかを決定することである。広範囲の種々の方法で組み合わせて、多くの異なる状況の要求を包含する適用範囲の広いテスト方法を与えるように命令を選択しなければならない。この要求を考慮すると、支援する自己テスト命令の数とそれらが表すテストの複雑さが増加するに従って、自己テスト・コントローラに関連する複雑さとオーバヘッドが増大する。

【0010】上記の要因を考慮して、本発明の好ましい実施の形態は次のテスト動作を行うことが可能な自己テスト命令を含む。すなわち、(i)　指定されたデータを或る範囲内のメモリ・アドレスの全てのメモリ位置に書き込み、(ii)　データを或る範囲内のメモリ・アドレスの全てのメモリ位置から読み取り、(iii)　指定されたデータをチェッカー盤パターンのメモリ・アドレスを有するメモリ位置に書き込み、(iv)　データをチェッカー盤パターンのメモリ・アドレスを有するメモリ位置から読み取り、(v)　マーチC(marchC)メモリ・テストを行い、(vi)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取りまた指定されたデータを書き込み、メモリ位置の或る行内のメモリ位置に順にアクセスした後でメモリ位置の次の行を選択してアクセスし、(vii)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取りまた指定されたデータを書き込み、メモリ位置の或る列内のメモリ位置に順にアクセスした後でメモリ位置の次の列を選択してアクセスし、(viii)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取り、指定されたデータを書き込み、またデータを読み取り、メモリ位置の或る行内のメモリ位置に順にアクセスした後でメモリ位置の次の行を選択してアクセスし、(ix)　メモリ位置の行と列に配列されたメモリ内の一連のメモリ位置からデータを読み取り、指定されたデータを書き込み、またデータを読み取り、メモリ位置の或る列内のメモリ位置に順にアクセスした後でメモリ位置の次の列を選択してアクセスし、(x)　一連のメモリ位置において、前記メモリ内の１つ以上のビット線に或る値を繰り返し書き込み、次に前記１つ以上のビット線を共用するメモリ位置内に記憶されている相補値を読み取り、(xi)　一連のメモリ位置において、或るメモリ位置から或る値を繰り返し読み取ると共に逆のデータ書込みを行い、(xii)　メーカのテスト方法が特定の要求を持たない実行／不実行(go/nogo)テストについて(i)から(xi)に規定されたメモリ・テスト動作の所定の組み合わせを行い、(xiii)　不合格検出を有効化するために特定の点で偽読取りデータ(false readdata)を生成する。

【0011】　本発明は、自己テスト・コントローラと１個以上のメモリとプロセッサ・コアとを１つの集積回路上に形成する実施の形態で用いるのに特に適している。同じ集積回路上に形成する比較的大量のメモリをプロセッサ・コアとして多く用いると、多数のテスト方法を支援する必要が増加する。なぜなら、大量のメモリを含む構成要素のメーカの数が増え、また異なる集積回路に与えられる製作特性の変動も増えるからである。別の観点からすると、メモリは、専門のメモリ・メーカが製作した離散的集積回路の形で与えられることは少なく、広範囲のメーカ（例えばＡＳＩＣメーカ）により種々の他の構成要素（特にプロセッサ・コア）と共に単一の集積回路内に含まれることの方が多いが、この方法はかかる状況における変化に対応する。

【0012】理解されるように、テストの対象となるメモリは種々の異なる形をとる。詳しく述べると、メモリは特定の環境に従って特注のメモリであったり合成メモリであったりする。これらの異なる種類は元来それぞれ異なる検査方法が必要であって、これには自己テスト・コントローラのプログラム可能な性質が有用である。

【0013】本発明の方法を更に複雑にする点は、異なるメモリを駆動するには異なる信号値とタイミングを必要とすることである。比較的簡単な設計の総称的自己テスト・コントローラを作ろうとすると、自己テスト・コントローラ内でかかる違いを処理するのは実用的でない。したがって本発明の好ましい実施の形態は、自己テスト・コントローラとテスト中のメモリの間の必要な信号値とタイミングの調整を行うインターフェース回路を含む。特定のメモリに接続するのに必要な回路要素をまとめて中間のインターフェース回路内に集中することにより、自己テスト・コントローラの設計を総称的にまた比較的コンパクトにできる。

【0014】用いるメモリ・アドレス・シーケンスを指定するだけでなく、自己テスト命令はテスト中にメモリに書き込むデータ値も指定する。データ値の選択は特定のテストにより検出される不良の種類に大きな影響を与え、またこれを用いて特定のメーカまたはユーザが設定できる方法でメモリの潜在的な弱点を探すことができる。

【００１５】理解されるように、最近の多くの集積回路は多数の異なるメモリを含む。これらの異なるメモリの設計は非常に異なり、異なるテスト方法を必要とする。自己テスト・コントローラはプログラム可能な性質を持つので、この単一回路によりかかる異なるメモリをそのメモリに最も適した方法でテストできる。また自己テスト命令は検出された誤りを報告する方法も指定する。これは優れた特徴であって、誤り報告を特定のメーカの要求に合わせ、または設計デバッグと製造テストとを比較するなど、異なるテスト環境を反映することもできる。

【００１６】自己テスト命令の制御の下にアクセス可能な記憶装置をインターフェース回路が備える場合は、検出された誤りに関する追加のデータを得ることができる。これにより自己テスト・コントローラとメモリの間のインターフェースが簡単になり、しかも特定のユーザまたは設計が望む場合は強力な診断またはその他のデータを収集できる。

【００１７】異なるキャッシュ・サイズやオンチップＲＡＭサイズなどの種々のメモリ・サイズの共通回路を効率的に処理するため、好ましい実施の形態では自己テスト命令はテスト対象のメモリのサイズを指定する。自己テスト・コントローラに関連する接続オーバヘッドを減らすため、好ましい実施の形態では自己テスト命令を自己テスト・コントローラに直列にロードする。自己テスト・コントローラは集積回路パッケージ上に複数の外部信号ピンを備えて、他の回路要素とは独立に製造テストを行うようにすることが望ましい。

【００１８】別の観点から、本発明はそれぞれのメモリ・アドレスに関連する複数のメモリ記憶位置を有するメモリのテスト方法を与える。この方法は、前記メモリに結合する自己テスト・コントローラに自己テスト命令を与え、前記自己テスト命令に応じて一連のメモリ記憶位置内の各メモリ位置に少なくとも一度メモリ・アクセスを行い、前記自己テスト命令により選択された前記一連のメモリ記憶位置内で順次にアクセスされるメモリ位置に従ってメモリ・アドレスは変化し、前記自己テスト・コントローラは前記自己テスト命令により形成されて異なるメモリ・テスト方法を実現する、ステップを含む。

【００１９】本発明の上記のまたはその他の目的と機能と利点は、例示の実施の形態の以下の詳細な説明を添付の図面と共に読むことにより明らかになる。

【００２０】

【発明の実施の形態】図１は、プロセッサ・コア４とキャッシュ・メモリ６とＲＡＭメモリ８（この例ではＳＤＲＡＭ）とを含む集積回路２を示す。理解されるように、かかる集積回路２は一般に更に多くの回路要素を含むが、簡単かつ明確にするために、それらの回路要素はこの図から省いた。

【００２１】集積回路２上に自己テスト・コントローラ１０があり、それぞれのディスパッチ回路１２、１４によりキャッシュ・メモリ６とＳＤＲＡＭメモリ８とそれぞれ通信する。これらのディスパッチ回路は自己テスト・コントローラ１０とそれぞれのメモリの間のインターフェース回路として作用する。これにより、自己テスト・コントローラが生成する総称的メモリ・テスト駆動信号を論理アドレスに再マッピングし、自己テスト・コントローラ１０の基本設計を変更せずに、異なるメモリの要求に対して値とタイミングを調整できる。

【００２２】また図から分かるように、自己テスト・コントローラ１０は集積回路２の多数の外部信号ピンの形で与えられる自己テスト・インターフェース１６を含む。このようにして、集積回路２の製作を完了すると適当なテスト・コントローラにプラグインして自己テスト・コントローラ１０の自己テスト活動を開始し、メモリ不良を検出すると集積回路２全体を除外するか、または可能であれば回路内で適当な修理を行う。

【００２３】図2はテスト・コントローラ１０とメモリ装置１８との関係をより詳細に示す略図である。自己テスト・コントローラ１０は自己テスト命令レジスタ２０を含み、直列シフト法を用いて自己テスト・インターフェース１６を介して自己テスト命令をこれにロードできる。自己テスト命令をロードすると、ＭＢＩＳＴ_ＲＵＮなどの所定の外部から与えられる信号をアサートすることにより、自己テスト動作が起動する。自己テスト・コントローラ１０内ではステートマシン２２が自己テスト命令のパターン選択部に応答する。パターン選択部は、自己テスト・コントローラ１０がどの物理的メモリ・アドレス・シーケンスを作成してメモリ装置１８に与えるかを選択する。

【００２４】例えば、選択されたパターンは次の活動を行う。指定されたデータ語を全ての項目に書き込み、全ての項目からデータ語を読み取り、チェッカー盤パターン内のデータ語または反転データ語を全ての項目に書き込み、チェッカー盤パターン内のデータ語または反転データ語を全ての項目から読み取り、マーチＣテストを行い（この種のテストはメモリ・テストの分野でよく知られているのでここではこれ以上を説明しない）、増分減分語線高速マーチを用いて、読取り動作を行い次に書込み動作を行い、増分減分ビット線高速マーチを用いて、読取り動作を行い次に書込み動作を行い、増分減分語線高速マーチを用いて、読取り動作と書込み動作と読取り動作を行い、増分減分ビット線高速マーチを用いて、読取り動作と書込み動作と読取り動作を行い、書込み動作と読取り動作の繰返しによりビット線対にストレス・テストを行う「バング」テストを行い、所定の一連のテスト（上に述べたテストなど）を行って実行／不実行の結果を生じ、不合格検出を有効化するために特定の点で偽読取りデータを生成するパターンを行う。

【００２５】上に述べた方法の一部として与えられるデータ語も自己テスト命令内のビット・フィールドとして

与えられる。自己テスト命令内のエンジン制御フィールドは自己テスト・コントローラ１０と自己テスト装置全体の誤り報告オプションを定義する。自己テスト命令の配列選択部は、多数のメモリ装置を含む環境において、関係する自己テスト命令でどのメモリ装置をテストするかを選択する。理解されるように、適当な環境では自己テスト命令を多数のメモリ装置に並列に与えることができる。

【００２６】自己テスト・コントローラ１０は、列アドレス信号と行アドレス信号の形の物理的メモリ・アドレス信号を生成してディスパッチ・コントローラ２４に送る。指定されたテストで与えられるデータ語もディスパッチ・コントローラ２４に送る。種々の他の制御信号も自己テスト・コントローラ１０とディスパッチ・コントローラ２４の間で交換される。

【００２７】ディスパッチ・コントローラ２４は総称的自己テスト・コントローラ１０が生成する信号に、必要な信号値調整またはタイミング調整を与えるインターフェース回路として動作する。これはメモリ装置１８内で所望のテストを行うのに必要である。例えば、受けるメモリ・アクセス命令と実行するメモリ・アクセス命令の間に異なる長さのパイプライン遅延を挿入することにより、異なる待ち時間の信号を与える。ディスパッチ・コントローラ２４の別の重要な役割は、自己テスト・コントローラ１０が生成した物理的メモリ・アドレス信号を、メモリ装置１８を駆動するのに必要な論理的メモリ・アドレス信号にマッピングすることである。ディスパッチ・コントローラ２４が特定のメモリ装置１８用に特に設計された場合は、このマッピングをその設計に含める。別の実施の形態では、ディスパッチ・コントローラ２４自体が或る程度総称的であって、必要に応じて異なるマッピングを処理できることが望ましい。これをオプションで行うには、物理的メモリ・アドレス信号をラッチし、これを更に別個に設計して設けられたマッピング回路２６に送った後、所望の論理的メモリ・アドレス信号を返して、ディスパッチ・コントローラ２４内でラッチする。この別個に設けられるマッピング回路２６を図２の点線内に示してこれがオプションであることを示す。理解されるように、マッピング回路はこのように別個に設けてもよいし、ディスパッチ・コントローラ２４内に組み込んでもよい。どちらの場合も本発明の方法に含まれる。

【００２８】ディスパッチ・コントローラ２４は、メモリ装置１８内でテストを行うのに必要な信号値を与える。これらの信号は、論理アドレス値と、データ値（一般に自己テスト命令内で指定されるデータ語の拡張形式である）と、任意の関連する制御信号を含む。またディスパッチ・コントローラ２４は、要求されたメモリ・アクセス動作に対するメモリ装置１８の応答を監視して、検出されたメモリ不良（読み返された不正なデータ・ビットなど）を自己テスト・コントローラ１０に報告する。ディスパッチ・コントローラ２４は、全ての検出された誤りまたはその他のパラメータを指定するデータをログするデータ・レジスタを含んでよい。このデータはテストが完了した後で、自己テスト・コントローラ１０に取込んだ当該自己テスト命令を用いてディスパッチ・コントローラ２４から外部の環境に読み取られる。この種の動作は、誤りの検出に関して細かいレベルの詳細が必要な場合の設計のデバッグに特に有用である。

【００２９】図3は、図１と図２に示す実施の形態の動作の概略を示す流れ図である。ステップ２８で、自己テスト命令を自己テスト・コントローラ１０に直列にロードする。ステップ３０で、図2に示すＭＢＩＳＴ_ＲＵＮ信号を外部から与えて、制御を自己テスト・コントローラ１０に渡す。ステップ３２で、自己テスト・コントローラ内のステートマシン２２は、自己テスト命令レジスタ２０内に記憶されている自己テスト命令からパターン選択ビットとエンジン制御ビットとデータ語ビットを読み取り、ステップ３４で、指定されたテスト信号を生成する。理解されるように、自己テスト命令は一般に、テスト中のメモリ装置に与えて所望のテスト方法の少なくとも一部を実行する何千もの物理的メモリ・アドレス信号値とデータ値を生成する長いシーケンスを指定する。

【００３０】特定の自己テスト命令の実行を完了した後、ステップ３６で、返した結果が合格か不合格かをテストをする。不合格の結果を返した場合は処理はステップ３８に進み、任意の所望の不合格活動をトリガする。結果が合格の場合はその自己テスト命令を直ぐ終了する。理解されるように、図３の流れ図は単一自己テスト命令の実行を示す。実際にはメーカまたはユーザは、特定の設計および製作環境に特に適したテスト方法を指定するのに必要な自己テスト命令の長いシーケンスの形でテスト方法を構築するのが普通である。このように多数の自己テスト命令を自己テスト・コントローラ１０にロードして、図3に示す処理を多数行う。パターン集合を制御するのにデータの保持や、外乱の読取りや、ｉｄｄｑテストを行う。理解されるように、ユーザによっては自分のテスト方法を特注で指定することを希望しないかも知れず、従って、単一自己テスト命令を用いて指定され単一の簡単な実行／不実行の結果値を返す上に述べた省略時テスト方法に依存するかも知れない。

【００３１】図4は自己テスト・コントローラ１０から物理的メモリ・アドレス信号４０を受ける方法を示す。これはマッピング回路４２が処理し、テストするメモリ装置を駆動する適当な論理的メモリ・アドレス信号４４を生成する。簡単な装置では、物理的メモリ・アドレス信号を論理的メモリ・アドレス信号に１対１にマッピングする。物理的メモリ・アドレス信号と論理的メモリ・アドレス信号の関係がより複雑な場合もあるが、それで

も設計が比較的簡単で容易なマッピング回路４２で達成できる。

【００３２】図5は提供されるアドレス信号マッピングの例を示す。テストするメモリ装置は２個のメモリ配列で形成し、それぞれ３２行ｘ８列を含む。論理アドレスは9ビット値である。最上位論理アドレス・ビットＬＡ［８］を用いてマルチプレクサを制御して、2個のメモリ配列のどちらかを選択する。論理アドレスの最下位３ビットＬＡ［２：０］を用いて列を選択する。残りの論理アドレス・ビットＬＡ［７：３］を用いて行を選択する。この関係と装置を図5の上部に示す。

【００３３】図5の中央部は図２と同じで、マクロセルを示す。マクロセルはＳＯＣ設計（自分のＳＯＣインターフェースを有する）の一部であり、また自己テスト・コントローラ１０とディスパッチ・コントローラ２４と上に説明した２ブロックで形成するメモリ装置を含む。自己テスト・コントローラ１０はメモリ装置を、５ビットの行アドレスＸａｄｄｒで表される３２行を有するメモリ位置の配列として処理する。メモリ位置の列は、この列が物理的装置内の低次ブロック内にあるか高次のブロック内にあるかに関わらず、４ビット・アドレスＹａｄｄｒで表される。図5の例では、マッピング回路はＳＯＣインターフェースに取り付けられた特注の回路を介してディスパッチ・コントローラ２４の外部に設けられる。特注のマッピング回路をこのレベルで取り付けることにより、この回路の変更と設計を容易に行うことができる。図に示す例では、物理的メモリ・アドレス信号をディスパッチ回路２４内にラッチし、次に別個のマッピング回路に送る。マッピング回路内では、Ｙａｄｄｒフィールドの最下位３ビットは論理アドレスの最下位３ビットＬＡ［２：０］を形成する。Ｙａｄｄｒ値の最上位ビットは論理アドレスの最上位ビットＬＡ［８］を形成する。行アドレス５ビット値Ｘａｄｄｒは論理アドレスのビットＬＡ［７：３］を形成する。このようにして構築された論理アドレスＬＡ［８：０］はマッピング回路からディスパッチ・コントローラ２４に戻されて再びラッチされた後、メモリ装置に与えられて所望のテストを行う。

【００３４】上に述べた方法の詳細は次の通りである。

内容

【００３５】範囲

この仕様は、メモリＢＩＳＴをハードおよびソフトのコア製品で実現したものの概要を示す。この仕様はＡＲＭ内の全ての新設計の青写真となるものであって、最終目的は再使用可能で一定のメモリＢＩＳＴ構造と、全ての顧客（ＡＲＭ内部を含む）用のメモリＢＩＳＴインターフェースを作成することである。

【００３６】概要

メモリＢＩＳＴ構造の機能性はハードおよびソフトのコア内に置く必要がある。なぜなら、メモリ・テスト・インターフェースを非クリティカル・パスに置くのがＡＲＭの希望だからである。この目的は、物理的配列からｎパイプライン段離れた書込み経路と読取り経路にアクセスすることにより達成される。配列テストの質を最高にするには配列の物理的マッピングが必要であり、またソフト・コアはアドレス・ポートを与えて、物理的形態が分かったときにこれに適合するように最終ユーザがアドレス・ピンを再設定できるようにする。

【００３７】ＡＲＭはオプションのメモリＢＩＳＴコントローラをソフト・コアの形で提供し、また特注の形をハード・コアで提供する（オプションではない）。メモリＢＩＳＴアルゴリズムに関して全ての最終ユーザが自分の考えを持っているのは明らかである。ＡＲＭの目的は、種々の不良の程度と種類をテストする標準および特注のアルゴリズムを提供することである。かかるパターンは全ての最終ユーザのテスト方法に完全に適合するわけではないが、最終ユーザが新しい方法を作成する必要がないほど十分な機能を有しているはずである。

【００３８】この仕様で注目すべきインターフェースが２つある。第1はユーザ・インターフェースである。ユーザ・インターフェースは将来の拡張を考えて指定する。このインターフェースによりロードされる主命令レジスタと同様に、どのコアでも使い方は同じである。こ

のためどの製品ラインでも見かけや感じが同じである。第2のインターフェースはメモリ・アクセス点に存在する。共通のインターフェースを持つ全ての将来のＡＲＭコアについて最終ユーザが自分のＢＩＳＴ方式を設計できるようにすると共に、当社の内部ＢＩＳＴエンジンの要求を満たすためには、種々のコアに共通のインターフェースを持つことが重要である。この文書は配列バイパスまたは制御などの特定のＡＴＰＧテストのニーズに応えるものではなく、ＡＴＰＧテスト可能性設計方式に関わらず必要な2つの機能を含むものである。

【００３９】ＭＢＩＳＴ構造

概要

ＡＲＭメモリＢＩＳＴ方式は物理的にマッピングされた配列でテストを行う機能を中心にして、パターンの有効性を高める。ハード・コアは必ずＸ次元（行／検出アンプとの間の距離）とＹ次元（列すなわち検出アンプの選択）に物理的にマッピングされる。ソフト・コアは、メモリ・テスト・インターフェースから、コアとメモリ・インターフェースで与えられる論理アドレスへの文書化されたマッピングを有する。これは、

１）　インターフェース毎にＢＩＳＴ＿ディスパッチ・ユニットをスクランブルすることにより物理的マッピングを行って目的のマッピングに合わせるか、

２）　コンパイルされたＲＡＭ内で強制的に物理的マッピングを行って所定のＡＲＭマッピングに合わせる、ためにインテグレータが実行すべきものである（テスト効率を最大にするため）。

【００４０】メモリＢＩＳＴ構造は、制御信号をコア内の全ての配列に分配する単一メモリＢＩＳＴテスト・エンジンを中心とする。コア毎に１つのメモリＢＩＳＴを作成すると仮定する。ただし、多数のコアにまたがってエンジンを統合することもできる。コア毎に２個のメモリＢＩＳＴエンジンを設けることは可能であるが、テスタとメモリＢＩＳＴコントローラとの間のテスト・インターフェースはシステム・インテグレータが管理し選択しなければならない。テスト信号の分配と経路選択の影響は、メモリＢＩＳＴコントローラから出る４ビットのデータ語バスを用いることにより最小になる。データ語の拡張とテストは、ｍＢＩＳＴ＿ディスパッチと呼ぶ特殊なユニットで行われる。これは、メモリＢＩＳＴコントローラとメモリＢＩＳＴテスト・インターフェースとの接着論理(glue logic)として働く。この接着論理は、アドレス・スクランブリング（物理的空間へのマッピング）と、特殊化されたパイプライニング（目標の配列に対してローカル）と、データ語の比較と、テスト不合格データロギングを行う。これにより、通常、集中化されたＢＩＳＴコントローラに関連する過大な経路選択オーバヘッドを大幅に減らすことができる。

【００４１】接着論理を含むディスパッチ・ユニットは設計に特有の形態を与え、しかもメモリ・テスト・インターフェースとＢＩＳＴコントローラを、どの設計でも使える固定された標準にすることができる。全ての将来のＡＲＭコアでそのまま再使用可能なＢＩＳＴコントローラを有することにより、どの設計においても最終ユーザ・テスト・インターフェースを標準化することができる。メモリＢＩＳＴは、メモリＢＩＳＴインターフェースへのアクセスを可能にするコアのテスト・インターフェースを形成することにより開始する。テスト・アルゴリズムの進行（アドレス満了フラグにより測定する）と合格／不合格状態は４ビットの出力専用データ・バス（マクロセルへのポートを有する）で連続的に送られる。更なるデータロギング情報は、ＢＩＳＴコントローラまたは不合格ユニットのディスパッチ・ユニットのシフト・アクセス（同じ外部ＢＩＳＴインターフェースを用いる）により得ることができる。ディスパッチ・ユニット内のデータロギングおよび追跡支援の量は設計に依存し、また設定可能である。

【００４２】各メモリＢＩＳＴインターフェースは多数の配列を支援することができる。これらの各配列はＢＩＳＴ命令レジスタと不良ログ内で別個の配列として支援される。テストの実行中は、これらの配列もそのイネーブル線をＹａｄｄｒ空間に追加することによりテストすることができる。命令レジスタのイネーブル線により可能になる場合は、ＢＩＳＴディスパッチ・ユニットはイネーブル線をより高いアドレス・ビットにマッピングすることができる。注意すべきであるが、これはこれらの目標配列の直列テストまたは並列テストの通常の定義とは一致せず、配列はより大きな全体配列のセグメントとして処理される。

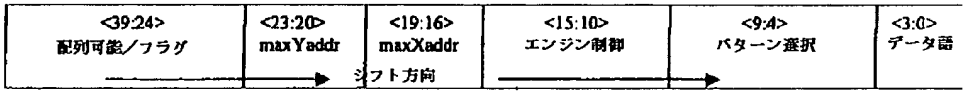
【００４３】図6は、集中化されたＢＩＳＴコントローラとディスパッチ・ユニットとを示す。ＭＢＩＳＴテスト・インターフェースの定義は次の表の通りである。注目すべきは、テスト構成をシフト・ローディングするためのアクセス機構である。このシフト・プロセスはコア・クロック領域で実行される。コアの動作が進行するに従って、低速で手動シフトを行うため（テスタ・インターフェースの制限により）のｐ１１切替え機構と、メモリ・テストを高速で行うため（クロック速度がテスタ・サイクルを超える場合、またはパッド・タイミング制限が最大クロッキング速度を抑える場合）の高速ｐ１１クロッキングへの切換えスイッチが必要である。将来の移行経路を支援するためにジャガー（Jaguar）用の４分割シフティング・クロックを形成すると良いかも知れない。

MBISTへのテスト・インターフェース		
コア・ポート信号	I/O、サイズ	説明
MBIST_RUN	In	この信号によりメモリ・テストを開始
MBIST_SHIFT	In	制御レジスタにデータのローディング可能
MBIST_DIN	In	制御レジスタへの直列ローディング
SCANMODE	In	ディスパッチ・ユニットでのオーバライド禁止
MBIST_DOUT	Out,<3:0>	ランタイム状態<実行、不合格、X満了、Y満了>

【００４４】３２クロック用のＭＢＩＳＴ＿ＳＨＩＦＴをアサートし、ＬＳＢを最初に、ＭＳＢを最後に送ることにより、テスト構成を３２ビット・レジスタにロードする。このレジスタ・ローディング中はＭＢＩＳＴ＿ＲＵＮを非活動状態にしなければならない。ＭＢＩＳＴ＿ＳＨＩＦＴをデアサート(de-asserted)にするとレジスタ・ローディングの実行が終了する。ＭＢＩＳＴ＿ＲＵＮをアサートすると選択されたメモリ・テストを開始し、テストの終了はＭＢＩＳＴ＿ＤＯＵＴ＜３＞ビットで知らせる。テスト中は、ＭＢＩＳＴ＿ＤＯＵＴ＜１：０＞はトグルしてアルゴリズムの進行をマークする。ＭＢＩＳＴ＿ＤＯＵＴ＜２＞すなわち不合格ビットはスティッキー(sticky)である。ＭＢＩＳＴ＿ＳＨＩＦＴはＭＢＩＳＴ＿ＲＵＮによりゲートされ、エンジン内の未知の危険を防ぐ。ＭＢＩＳＴ＿ＤＯＵＴ＜０＞は（ＭＢＩＳＴ＿ＳＨＩＦＴ　&～　ＭＢＩＳＴ＿ＲＵＮ）のとき命令レジスタのシフト・データ・アウト経路を兼ねる。

【００４５】ＭＢＩＳＴ命令レジスタ

概要：



【００４６】配列イネーブルとフラグ＜３９：２４＞

コア内のテストされる各配列はＴＲＭ文書の指定に従ってイネーブル・ビットが割り当てられる。これは、全ての配列を並列にテストするか、必要に応じて柔軟に配列を個別にテストするかを示す。例えば、最大パワーの関係からパス当たり５０％の配列だけテストする場合がある。別の例では、ＡＴＰＧまたはｉｄｄｑで用いるために種々の配列を初期化して種々の背景パターンにする。配列はメモリ・テスト・インターフェースの数で定義されるのではなく、個別にテストされる配列の数により定義される（単一メモリ・テスト･インターフェースは多数の配列イネーブルを有してもよい）。命令レジスタのこの部分は第２の目的も有する。外部のユーザには外部インターフェース上に単一の不良フラグだけが見える。レジスタのこの部分を走査すれば各配列のＰ／Ｆ状態が分かり、不良配列を更に識別することができる。

【００４７】ｍａｘＹａｄｄｒ＜２３：２０＞

ｍａｘＹａｄｄｒフィールドにより、ユーザはテスト中に用いる最大Ｙａｄｄｒ（列アドレスともいう）を指定することができる。種々のキャッシュ・サイズのコア・デリバティブを用いるとき、またはコア内の異なるサイズの配列をテストするとき、これは特に有用である。このＢＩＳＴコントローラ構造は異なるサイズの多数の配列を同時にテストすることを妨げない。このようなテストを行うには、ユーザは最大配列に等しい最大アドレス・サイズをプログラムし、ディスパッチ・ユニットは小さな配列の範囲外アドレスを阻止する（ＲＥ，ＷＥをゲートすることにより）。次の表はレジスタ設定と、その結果のアドレス・サイズを示す。

MaxYaddr	サイズ	BistYaddr活動ビット[11:0]
0000	0	000000000000
0001	2	000000000001
0010	4	000000000011
0011	8	000000000111
0100	16	000000001111
0101	32	000000011111
0110	64	000000111111
0111	128	000001111111
1000	256	000011111111
1001	512	000111111111
1010	1k	001111111111
1011	2k	011111111111
1100	4k	111111111111
>1100	--	支援されない

【００４８】ｍａｘＸａｄｄｒ＜１９：１６＞

Ｙアドレスと同様に、Ｘアドレス・フィールドはアドレス指定可能な行の最大数を選択する。ビット線ＲＣのために、５１２を超える行の深さは通常は実用的でないが、ユーザがトポロジ内の平面特性をＸアドレス空間にマッピングしたい場合は、ここでは最大２ｋの空間が支援される（平面すなわちブロック・アドレスはＹアドレス空間の上にマッピングすることを勧める。さもない

と、Xアドレス空間に基づくビット線ストレス・テスト　　の効率が悪くなる)。

MaxXaddr	サイズ	BistXaddr活動ビット[10:0]
0000	0	00000000000
0001	2	00000000001
0010	4	00000000011
0011	8	00000000111
0100	16	00000001111
0101	32	00000011111
0110	64	00000111111
0111	128	00001111111
1000	256	00011111111
1001	512	00111111111
1010	1k	01111111111
1011	2k	11111111111
11xx	--	支援されない

【0049】エンジン制御<15:10>　　シンの行動(不良で停止、など)を指定する。<15:
レジスタのエンジン制御部を用いて、特殊なステートマ　　10>=0という設定は省略時行動を指定する。

<15:10>	行動	説明
00 0000	デフォールト	テスト実行の終了
1x xxxx	実時間不合格ビット	不合格ビットは実時間、スティッキーな不合格がテストの終わりに出現
xx 0001	不合格で停止	16サイクル待ち時間未満の不合格で停止
xx 0010	不合格ならアルゴリズムの終了で停止	現在のアリゴリズム「N」が完了後の不合格で停止
00 1110	ディスパッチ・ユニット命令のロード	1ホット<31:16>で選択されたディスパッチ・ユニットを走査
00 1111	ディスパッチ・ユニット・ダイアログの読取り	1ホット<31:16>で選択されたディスパッチ・ユニットを走査

【0050】「実時間不合格ビット」は、テスト/試験室要員がソフト不合格行動を監視したいときに試験室で用いるためのものである。「アルゴリズム・パスの終了で停止」は、不合格アルゴリズム・セグメントの終了までテストを進行させる手段として与えられる。全ての配列項目をダウンロードするためにディスパッチ・ユニットが走査可能なテスト・アクセス機構を備える場合、これはビット・マッピングに有用である。

【0051】ディスパッチ・ユニットを走査すると、選択されたディスパッチ・ユニット走査ポートはMBIST_RUNがアサートされたときにMBIST_DINとMBIST_DOUTの間に置かれる。MBIST_RUNがデアサートになるとアクセス禁止になる。ディスパッチ・ユニットにロードされた全ての動作はMBIST_RUNのデアサートのときに実行される。(注意:　効率を最大にするにはハンドシェイクの定義を改善する必要があり、またmBIST_RUNなどの特有の機能は暖機サイクルを必要とするので動作を完了するのに十分な時間を与える必要があり、適当なサイクル・カウントだけ内部オーバライド・リリースを遅らせれば恐らく十分である)。ディスパッチ・ユニットはこのとき緩く定義されており、コアの必要に応じて構築される。ディスパッチ・ユニットは、最初の不合格アドレスや不合格ビットなどの最小のデータロギング情報を与えなければならない。この情報は、ディスパッチ・データログを走査することによりアクセス可能でなければならない。走査可能なディスパッチ・ユニット命令の使用はオプションである。

【0052】パターン/アルゴリズム選択<9:4>
多くの標準パターンとARMが作成したビット線ストレス・パターンがテスト用に設けられている。未知の最終テスト設計で最も柔軟に対応できるように、アルゴリズムは断片的に作成されている。支援されるテストの領域は、実行/不実行、iddq、データ保持、ATPG、総合機能テスト、ストレス・テストなどである。次の表は支援されるパターンを示し、後のパラグラフはその使用法を示す。セグメント化されたテストを用いれば、工場内でのsoftbinおよび歩留まり追跡情報を最大にし、最終テスト設計を工場のメモリ・テストの考え方に最も良く合わせることができる。

<5:0>選択	名前	「N」	説明
00000	WriteSolids	1N	指定されたデータ語を全ての項目に書き込む
00001	ReadSolids	1N	全ての項目からデータ語を読み取る
00010	WriteCkbd	1N	X<0>^Y<0>で選択されたデータまたはデータバー
00011	ReadCkbd	1N	X<0>^Y<0>で選択されたデータまたはデータバー
00100	March C+	14N	増分減分XY高速RWRマーチ
00101	PttnFail	6N	BIST不良検出をテストする
00110	RW_Xmarch	6N	増分減分語線高速マーチ
00111	RW_Ymarch	6N	増分減分ビット線高速マーチ
01000	RWR_Xmarch	8N	増分減分語線高速マーチ
01001	RWR_Ymarch	8N	増分減分ビット線高速マーチ
01010	Bang	18N	増分減分語線高速ビット線ストレス
11111	Go/No Go	30N	2*W/RCkbd、RWR Ymarch, X Bang

【0053】

定義：

列	ビット線に並列の配列の次元（同じSenseAMP上）
行	語線に並列の配列の次元
行高速／Xfast	目標セルは次の列に移る前にビット線に沿って動く
列高速／Yfast	目標セルは行／語線の前にビット線対をまたがって動く
Xfast増分	目標セルが最も近い検出アンプから離れる
Xfast減分	目標セルが検出アンプから最も遠い点から近づく
Yfast増分	Yaddr空間が0から最大まで、東西方向に動く
Xfast減分	Yaddr空間が最大から0まで、増分の逆に動く
N	アドレス指定可能な項目の数

【0054】BISTパターン仕様：全てのメモリBISTテストは物理的にマッピングされたアドレス空間に対して実行される。これは単に、最下位Xaddrが隣接する行の間で切り替わることを意味する（LSB+1は2行置きに切り替わる、など）。Yaddr空間も物理的にマッピングされる。これにより、供給されたパターンでメモリ不良を効率的にまたは直接探すことができる。「writeCkbd」は実行されたXfastである。このパターンは1Nで、書込み専用である。データ極性はxor(Xaddr[0],Yaddr[0])で設定される。「ReadCkbd」は実行されたXfastである。このパターンは1Nで、読取り専用である。データ極性はxor(Xaddr[0],Yaddr[0])で設定される。「writeSolids」は実行されたXfastである。このパターンは1Nで、書込み専用である。データ極性＝真。「ReadSolids」は実行されたXfastである。このパターンは1Nで、読取り専用である。データ極性＝真。

【0055】「MarchC+」は次のシーケンスで実行されたXfastである。

1) write solids（配列初期化）incr
2) Rdata, Wdatabar, Rdatabar incr
3) Rdatabar, Wdata, Rdata incr
4) Rdata, Wdatabar, Rdatabar decr
5) Rdatabar, Wdata, Rdata decr
6) Rdata Verify decr

「RWR_Xmarch」は次のシーケンスで実行されたXfastである。

1) Write solid（配列を初期化）
2) Rdata/Wdatabar/Rdatabar inc
3) Rdatabar/Wdata/Rdata decrement
4) Rdata solid

「RWR_Ymarch」はRWR_Xmarchと同じシーケンスを有するが実行されたYfastである。

【0056】「RW_Xmarch」は実行されたXfastである。この6Nパターンは次のシーケンスを有する。

1) Write Solid（配列を初期化）
2) Rdata/Wdatabar inc
3) Rdatabar/Wdata decrement
4) Rdata solid

「RW_Ymarch」は6Nで、RW_Xmarchと同じシーケンスで実行されたYfastである。「pttnFail」は実行されたXfastである。WriteSolidパターンを実行し、次にReadSolidを実行する。不合格は、ReadSolid中にアドレス選択でデータ極性を逆にして注入する。このパターンは、BIST検出論理（目標の配列での）が機能することを確認するのに必要である。

【0057】「bang」は18Nで、実行されたXfastである。ビット線対上で多数の書込みと読取りを連続して実行する。
1)write solid(配列を初期化)、
* 2)read data target,write databartarget,6*write databar row0,increment to next target Xfast
** 3)5*read databar target,write datarow0,read databar target,write data target
4)read data(配列を確認)。
*このセグメントはビット線対を「バング」して、書込み後に正しい等化を保証する。等化/プレチャージが不十分な場合は、同じビット線対から逆のデータを読み取るときに読取りが遅くなる。自己計時するキャッシュでの読取りが遅いと、マーチCのようによく使われる「シングル・ショット」に見られない機能不良が発生する。
**このセグメントはビットセルを歩き、そのビット線対に逆のデータを書き込み、目標セルのデータを読み取るためのものである。この不良機構は6TのRAMセルでは余り用いられない(4TやDRAMに比べて)。
*, **「犠牲的(sacrificial)」行を用いると、グレーコード・パターン・シーケンスがないときに、Xaddr空間内のオープン・デコーダ不良を検出するのに役立つ(デコード・ブロックが小さいのでYaddrは構造的に不良の種類に入らない)。
*, **このパターンは不良のスタックも検出するが、その主な目的はメモリのアナログ特性をアドレス指定することである。

【0058】「実行/不実行」、30Nは、最終ユーザが自分のテスト方式を作成することを望まない場合にメモリ・テスト用に選択されるARMテスト・パッケージである。このパターンを選択すると、その後のテストは次に示すリストに従って実行する。これはその配列の合格/不合格状態を最終ユーザに示す。次に示すテスト・パッケージは配列の総合的機能テストを行う(データ保持や他のストレス・テストは含まない)。実行/不実行の一連のテストはメモリ・テストにおける社内のARMメモリ・テスト技術者の過去の経験の集大成である。
1) Perform wckbd 5's
2) Rckbd 5's
3) Wckbd a's
4) Rckbd a's
5) RWR_Ymarch 6's
6) Bang 0's
【0059】データ語<3:0>
テスト命令をロードするとき、ユーザはテスト・アルゴリズム中に用いられるルート・データ語を指定する(実行/不実行BISTアルゴリズム中は除く)。これにより、ユーザは初期化された配列内に記憶されている値(iddqまたはATPG)を選択し、または新しいデータ語を選択してマーチおよびビット線ストレス・テスト中の予期しない感度を探すことができる。
【0060】MBISTメモリ・インターフェース
メモリ・インターフェースの定義を下の表に示す。ARMがそのコア用のメモリを提供するかどうかに関わらず、コア上の全てのメモリ装置でこれを用いる必要がある。これにより、ユーザはARMのBISTコントローラを特に修正せずに用いることができる。ATPG関連の支援もここに定義されている。

信号	I/O、サイズ	説明
MTestSel	In,<?:0>	オプション、多数の下流の配列の直列テストに用いる
MTestWEx	In	書込み動作を命じる
MTestREx	In	読取り動作を命じる
MTestADDR*	In,<maxAddr:0>	目標アドレス
MTestDINx	In,<datawidth:0>	BIST書込みデータ
MTestDOUTy	Out,<datawidth:0>	BIST読取りデータ
MTestON	In	キャッシュ論理のグローバル・オーバライド
ArrayDisable	In	動作モードに関わらず、全ての配列の活動を阻止する

*関係する全てのスクランブリングをmBIST_ディスパッチが処理する場合は、単一アドレス・バスを有する。
【0061】メモリBISTインターフェースの定義
「MTestSel<?:0>」
メモリBISTインターフェースが2つ以上の下流の配列を支援する場合は、目標とする配列を選択するのに別個のイネーブルが必要である。この信号は多数の修理可能な領域を有する単一配列にも必要である。単一配列インターフェースはこの信号の使用またはポートを必要としない。

「MTestWEx」
この信号は(MTestSelまたはMTestONにより内部で阻止されない場合は)メモリ・テスト配列インターフェースに書込み動作を実施中であることを示す。アドレスおよびデータ情報はこの信号と同時に送られる。書込み動作を完了するのに2サイクルかかる場合は、この信号を「MTestWE2」で表す。多数の配列がそのインターフェースで支援される場合(全てが異なるパイプライン遅延を有する)はxを用いる(「MTestWEx」)。信号名の中にパイプライン遅延の長さを用いることにより、BISTインターフェース・ユ

ニットまたは最終ユーザ自身によるＢＩＳＴ統合が容易になる。

【００６２】「ＭＴｅｓｔＲＥｘ」

テスト中の配列の読取りイネーブルである。詳細はＭＴｅｓｔＷＥｘ信号を参照のこと。

「ＭＴｅｓｔＡＤＤＲ＜？：０＞*」

テスト中の配列内の目標アドレスである。信号の広さは目標配列内のアドレス指定可能な要素の最大数と同じである。

【００６３】「ＭＴｅｓｔＤＩＮｘ，ＭＴｅｓｔＤＯＵＴｙ」

ＢＩＳＴ動作のためのデータイン・ポートとデータアウト・ポートである。ＭＢＩＳＴＤＯＵＴは、ＭＢＩＳＴＲＥｘの定義に従って「ｘ＋ｙ」サイクル後に送られる。例：ＭＢＩＳＴＲＥ３は、ＭＢＩＳＴＲＥ３情報がアサートされてから３クロック後に読取り動作を行うことを指定する。データアウト・パイプラインは自身のオフセットを有し、これを「ｙ」で表す。多数の配列が単一のインターフェースで支援され、またいくつかの配列のデータ幅が最大配列幅より小さい場合は、全ての読取り動作において全ての未使用データ・ビットで「０」を送ることがメモリＢＩＳＴインターフェースに要求される。データイン・バスとデータアウト・バスが下流の配列に依存して種々のパイプライン・オフセットを有する場合は、「ｘ」を用いてオフセットを指定しなければならない。

【００６４】「ＭＴｅｓｔＯＮ」

これはグローバル・オーバライド信号で、ＢＩＳＴ動作が直ぐ起こることをインターフェースと下流の論理に知らせる。この信号は、キャッシュ論理内でＳｃａｎＭｏｄｅ／ＳｃａｎＥｎに次いで２番目に優先度の高い信号である。正規関数／ミッション・モード動作の優先度は更に低い。

「ＡｒｒａｙＤｉｓａｂｌｅ」

ＡＴＰＧ信号であって、これが活動状態のときは全ての配列への書込みと読取りを阻止する。配列の初期化データの安全を保証するため、阻止は全ての走査可能な状態要素の後に行わなければならない。この信号はメモリＢＩＳＴディスパッチ・ユニット内で生成され、ＩＤＤＱｔｅｓｔとＳｃａｎＥｎの「論理和」関数である。ＩＤＤＱｔｅｓｔはｉｄｄｑ走査および収集中に状態を保存するのに用いられ、ＳｃａｎＥｎはＡＴＰＧのＲＡＭテストを行うときにＡＴＰＧロード／アンロード中の状態を保存するのに用いられる。*関係する全てのスクランブリングをｍＢＩＳＴ＿ディスパッチが処理する場合には、単一アドレス・バスを有する。これは研究中である。

【００６５】ＭＢＩＳＴインターフェース設計規則

・　ＭＴｅｓｔＯＮは全ての正規関数動作をオーバライドする。

・　ＭＴｅｓｔＯＮは最初の動作の少なくとも４サイクル前にアサートされ、最後の動作が完了するまで解除されない。ＢＩＳＴテスト期間中は［オン］のままである。

・　装置モード（関数、ＡＴＰＧ、ＢＩＳＴなど）に関わらず、ＡｒｒａｙＤｉｓａｂｌｅは全ての書込み／読取り活動を阻止し、最後の操作可能な状態要素の後に起こる。

・　ＢＩＳＴノーオペレーションが起こることがある。ＲＥ／ＷＥが共に非活動状態のときのＢＩＳＴサイクルとして定義される。

・　バック・ツー・バック動作を支援しなければならない。これはＲ／Ｗ／Ａｄｄｒの任意の組合わせで予想される。

・　Ｆ／Ｆから与えられる全ての動作データと、読取りデータとは、Ｆ／Ｆ内にサンプリングされる（ｍＢＩＳＴはフロップに基づく設計である）。

・　全てのＢＩＳＴ動作は最高速度で起こり得る。

・　Ａｄｄｒ信号とＲＥ／ＷＥ信号は同じクロック・サイクルで与えられる。必要であればデータインは異なる段階で注入してよい。

・　全てのＲＥ／ＷＥ／Ｄａｔａ信号は信号記述で指定されたサイクル接尾部指示子を有しなければならない。

・　全ての読取り動作はＲＥアサートからデータアウトまで８サイクル以内に完了しなければならない（パイプライン注入点は妥当な点でなければならない）。

【００６６】・　多数の配列がインターフェースで支援される場合は、アドレスとデータの調整はディスパッチ・ユニットで行う（例：　アドレス指定された配列はＢＩＳＴ＿エンジンより小さい……ＢＩＳＴ＿ディスパッチは範囲外のアドレスについてＲＥ／ＷＥを阻止する）。

・　最も右のビットがＬＳＢである到着ＢＩＳＴＡＤＤＲにｗｒｔをアドレス指定するときは、全ての可変サイズの配列を右に整えなければならない。

・　動作の設定／保持時間に関係なく、ＡｒｒａｙＤｉｓａｂｌｅとＭＴｅｓｔＯＮだけをトグルすると仮定してよい。インターフェースでの全ての他の制御信号は各サイクルで未知と考えなければならない。

・　ｍＢＩＳＴインターフェースと実際の配列インターフェースの間でアドレス

・スクランブリングを行ってはならない。スクランブリングはディスパッチ・ユニットで行う。

・　各特有の配列インターフェースの下流の全ての配列で物理的マッピング特性を保持しなければならない（配列が所定のインターフェースで支援されるかどうかに関わらず、ＸａｄｄｒのＬＳＢは常に隣接行の間で選択する）。

【００６７】ＭＢＩＳＴディスパッチ・ユニット

各ＢＩＳＴコントローラとメモリ・テスト・インターフ

ェースの間にディスパッチ・ユニットが必要である。このブロックは、標準化されたフロントエンドＢＩＳＴコントローラと標準化されたメモリ・テスト･インターフェースの間の接着論理として働く。また配列への手動デバッグ・アクセスに用いられ、またテスト･インターフェースから読み取ったデータの比較とデータロギングを行う。

【００６８】ディスパッチ・ユニット・インターフェース

次の表はｍＢＩＳＴ_ディスパッチ・ユニットの入力および出力インターフェース信号を示す。

メモリＢＩＳＴコントローラからの入力／Ｏデータ		メモリ・テスト・インターフェースへのＩ／Ｏ出力	
信号	説明	信号	説明
ArrayEn<?:0>	目標配列可能(命令レジスタ)	MTestSel	インターフェースへの配列可能
BISTTest	ゲーム・オン	MTestWEx	配列書込み可能
MBISTData<3:0>	データの書込み／読取り	MTestREx	配列読取り可能
MBISTXaddr<?:0>	ＢＩＳＴコントローラ語線アドレス	MTestADDR*	配列セル選択
MBISTYaddr<?:0>	ＢＩＳＴコントローラ・ビット線アドレス		
MBISTWE	ＢＩＳＴコントローラ書込み可能	MTestDINx	配列データイン(全幅)
MBISTRE	ＢＩＳＴコントローラ読取り可能	MTestDOUTy	配列データアウト(全幅)
MBISTShiftJ	ディスパッチ命令のシフト可能	MTestON	配列テスト・オーバライド
MBISTShiftD	データログのシフト可能	ArrayDisable	配列アクセス阻止
DispatchDin	シフト・データイン		
DispatchDout	シフト・データアウト		
IddqTest	ｉｄｄｑテスト・モード		
SE	走査可能入力		
SCANMODE	ＡＴＰＧテスト・モード		
MBISTFail<?:0>	主コントローラに送られる不合格ビット		

【００６９】注：

１．　ＢＩＳＴＤａｔａ＜３：０＞は全メモリ･ビット幅にわたって直線的に拡張される。ディスパッチ・ユニット内でのこのデータのパイプライニングは＜３：０＞として行われ、必要に応じて拡張される。この拡張語を用いて、テスト中のメモリの全バス幅を駆動しまたは比較する。

２．　ＤｉｓｐａｔｃｈＤｏｕｔはディスパッチ･ユニットの命令レジスタまたはデータログ･レジスタのＬＳＢに続く（シフト・イネーブルにより選択される）。

３．　全ての定義された信号について、メモリ・テスト･インターフェースはディスパッチ･ユニットで、そしてディスパッチ･ユニットだけで駆動しなければならない（ＢＩＳＴコントローラに直接接続しない。なぜなら、ディスパッチ･ユニットはタイミングの安全のためにコントローラ･データをパイプラインするからである）。

４．　ディスパッチ内の命令レジスタ機能の支援は、設計に特有である。データロギング／修理の情報はディスパッチ･ユニット内で生成され保持され、設計に特有である。

５．　多数のイネーブルが選択した配列は、イネーブルを高次のＹａｄｄｒにマッピングすることにより同時にテストすることができる。これは実行／不実行のテストに非常に望ましい。

【００７０】アドレス・スクランブリング

ハード･コアは既知の物理アドレス空間から論理アドレス空間へのマッピングを有し、これに従ってＢＩＳＴ_ディスパッチ･ユニット内でスクランブルする必要がある。これはディスパッチ･ユニットで行う。なぜなら、アドレス･スクランブルの要求はメモリ・テスト･インターフェースによって異なるからである。ソフト・コアとコンパイルされたＲＡＭは既知のアドレス指定アップフロントを有しないので、特にメモリをコンパイルする前にコア設計を合成する場合は、スクランブリングをｒｔｌで行うことはできない。スクランブリングは、物理的マッピングがＡＲＭのＢＩＳＴアルゴリズムと共に存在することを確認する必要がある。これは、ＴｅｓｔＡｄｄｒの2つの追加のポートをディスパッチ・ユニットに加えることにより行う。１つはＢＩＳＴコントローラ･アドレスを表し、もう１つはＭＴｅｓｔＡｄｄｒインターフェースに行くアドレスのスクランブルされたものを表す。これらのポートは設計の最高ハード化レベルに上げ、必要に応じてコントローラ･アドレスをテスト・アドレスにステッチする(stitching)ことによりスクランブリングを行う。この組込みはコア･インテグレータが行う。

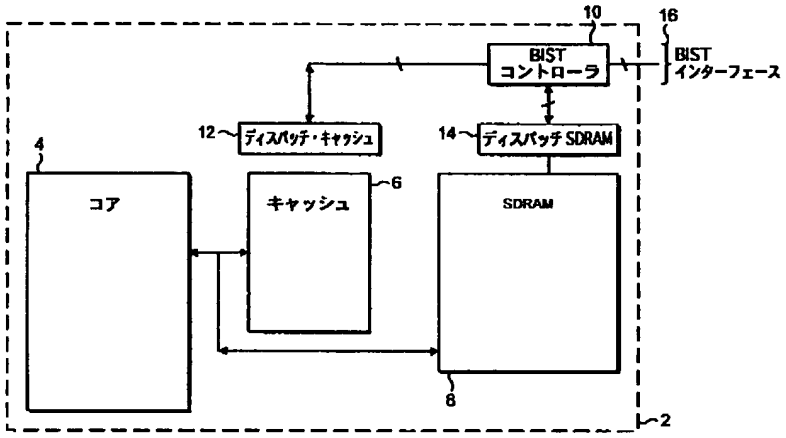
【００７１】添付の図面を参照して本発明の例示の実施の形態を詳細に説明したが、理解されるように、本発明はこれらの実施の形態に制限されるものではなく、当業者は特許請求の範囲に規定されている本発明の範囲と精神から逸れることなく種々の変更や修正を行うことができる。

【図面の簡単な説明】

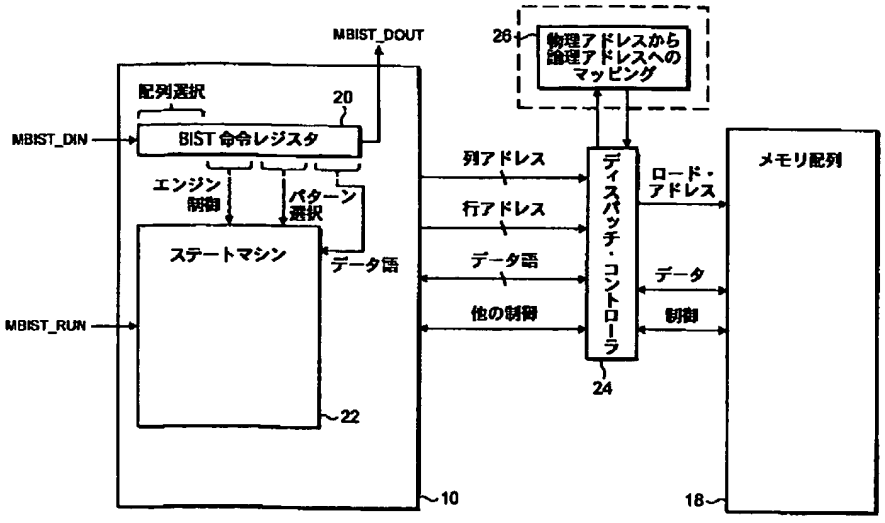
【図１】プロセッサ･コアと多数のメモリと自己テスト

・コントローラを組み込んだ集積回路の略図。
【図2】自己テスト・コントローラとインターフェース回路とメモリの詳細の略図。
【図3】図2の自己テスト・コントローラの使用の概要を示す流れ図。
【図4】物理的メモリ・アドレス信号から論理的メモリ・アドレス信号へのマッピングの概要を示す図。
【図5】必要なアドレス信号マッピングの例の詳細を示す図。
【図6】集中化されたＢＩＳＴコントローラとディスパッチ・ユニットとを示す図。
【符号の説明】
4　プロセッサ・コア
6　キャッシュ・メモリ
8　RAMメモリ
１０　自己テスト・コントローラ

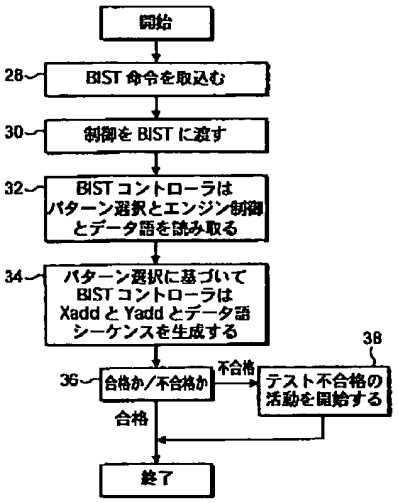
【図1】



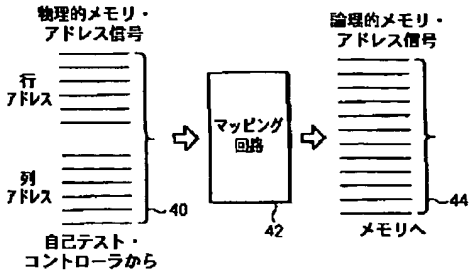
【図2】



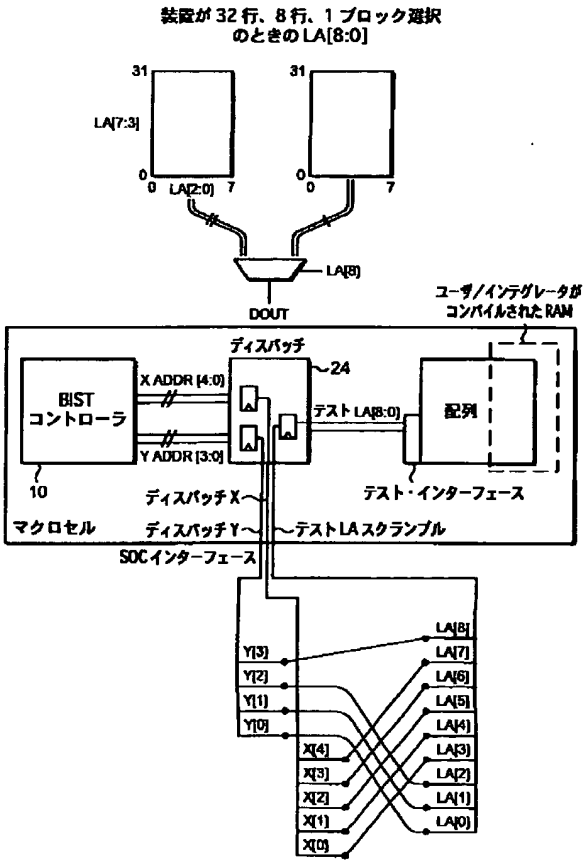
【図3】


開始
28
BIST 命令を取込む
30
制御を BIST に渡す
32
BIST コントローラは
パターン選択とエンジン制御
とデータ語を読み取る
34
パターン選択に基づいて
BIST コントローラは
Xadd と Yadd とデータ語
シーケンスを生成する
38
36
合格か/不合格か
不合格
テスト不合格の
活動を開始する
合格
終了

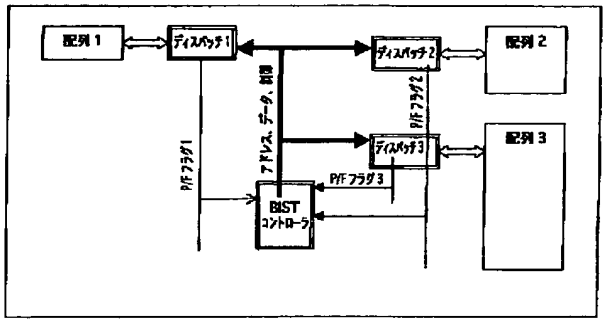
【図4】


物理的メモリ・
アドレス信号
行
アドレス
列
アドレス
40
自己テスト・
コントローラから
マッピング
回路
42
論理的メモリ・
アドレス信号
44
メモリへ

【図5】


装置が 32 行、8 行、1 ブロック選択
のときの LA[8:0]
31
LA[7:3]
0
0
LA[2:0]
7
31
0
0
7
LA[8]
DOUT
ユーザ/インテグレータが
コンパイルされた RAM
ディスパッチ
24
BIST
コントローラ
X ADDR [4:0]
Y ADDR [3:0]
テスト LA[8:0]
配列
10
テスト・インターフェース
マクロセル
ディスパッチ X
ディスパッチ Y
テスト LA スクランブル
SOC インターフェース
Y[3]
Y[2]
Y[1]
Y[0]
X[4]
X[3]
X[2]
X[1]
X[0]
LA[8]
LA[7]
LA[6]
LA[5]
LA[4]
LA[3]
LA[2]
LA[1]
LA[0]

【図6】


配列 1
ディスパッチ 1
ディスパッチ 2
配列 2
アドレス、データ、制御
P/F フラグ 2
P/F フラグ 1
ディスパッチ 3
配列 3
P/F フラグ 3
BIST
コントローラ

フロントページの続き

(72)発明者 スティーブン ジョン ヒル
アメリカ合衆国 テキサス、オースティン、 サウスウェスト パークウェイ 5604 ナンバー 2234

(72)発明者 ジェラード リチャード ウィリアムス ザ サード
アメリカ合衆国 テキサス、サンセット ヴァリー、 カーリー メスキート コーブ 3

Fターム(参考) 2G132 AA08 AK07 AK13 AK29
5B018 GA03 HA01 JA21 JA22 NA02
5B062 CC01 DD10 JJ05
5L106 AA16 DD22 DD23 EE02 GG07